Discrete traffic data generation using ML methods

CS-433 Machine learning – Project 2

Luca Bataillard, Julian Blackwell, Changling Li
Supervised by Tim Hillel, TRANSP-OR lab, EPFL

Abstract

This paper introduces a way to generate synthetic discrete traffic data for use in traffic simulators. We present a model capable of taking historical traffic data from a given week able to predict the discrete arrival times of vehicles in the same week, one year later. This is achieved using two phases. In the first phase, a recurrent neural network predicts the arrival rate for each hour in the week. In the second phase, a custom discretization model decomposes our rates into discrete arrival times. Our model outperforms the naive baseline approach in predicting arrival rates, and is successful in emulating real-world arrival time patterns and entropy.

I. Introduction

Traffic simulators like SUMO need vehicle data to perform modelling tasks. These simulators are widely used in the fields of civil engineering and urban planning for modelling complex traffic systems. They can provide city-wide traffic forecasts, allow for the evaluation of novel traffic control systems, or can even model the effect of autonomous vehicles on traffic patterns. [1]

To simulate traffic flow, simulators will generate simulated vehicles on a given road based on a log of the speed and arrival time of vehicles on that road. Such real world data is obtained using a car sensor placed on or near the roadway. These sensors are not always online: they can break down or can be moved to be used elsewhere. They also cannot predict future traffic trends. This means that the logs sometimes have large gaps if using real-world data.

In this paper, we present a way to fill in the gaps in traffic logs and to predict future traffic logs up to a year in advance, using machine leaning models. Our model takes a week-long traffic log and outputs a traffic log for the same week one year later. It can be combined to predict an entire year in parallel using data from the previous year.

The model we propose has two phases, a rate prediction phase and a discrete event generation phase. In the first phase, we predict a vehicle arrival rate for every hour in the week, using a LSTM recurrent neural network [2]. The second phase takes the hourly rates and outputs a discrete arrival log. It first predicts a 5-minute interval arrival rate using historical averages, then samples a Poisson distribution to generate discrete events.

II. Exploratory data analysis

We first start with some exploratory data analysis to better understand our time series data before further processing. Our dataset contains a variety of features but we mainly focus on the weight, speed, and crossing times of each vehicle.

A. Initial exploratory data analysis

Our data ranges over a period of almost 10 years from 2011-04-01 to 2021-03-28. The minimum measured weight is 3.5t, which means that the vehicles are mainly heavy trucks. Indeed, the maximum recorded weight reaches 99.9t. 90% of all observations lie between 3.5 and 40 tons.

Some repetitive trends can be noticed. There is less traffic during the nights and the speed drops significantly around 5pm during the workdays (likely due to traffic jams). We also notice that the average speed is relatively constant around 100km/h from 6am to 5pm (the speed limit [3]). Many outliers do still exist, notably there is a highest recorded speed of 255 km/h.

B. Missing data points and further analysis on weight, speed and time

We notice some periods of time with missing data points, especially towards the start of the COVID pandemic. After some exploration of the monthly number of vehicles per year (see Fig.1), there are also entire months with no data particularly in 2013 and 2014.

Next we observe that a large majority of vehicles are bunched together between 65 and 100 km/h. After a quick look on the weight distribution among the vehicles, the weight is heavily left-tailed, with some rare exceptions weighing more than 50 tons (see fig. 2). We also find that the mean weight approximately varies between 16-17 tons each month.

After analysing a possible correlation between weight and speed, at first it does not seem like there is any significant dependence between speed and weight of a vehicle. However, there is a higher variance in speed for lower weights (see 3).

The number of vehicles evolves depending on the day of the week. There is a big drop on Saturday, and even more on Sunday. Weekdays are quite uniform, except for Monday and Friday who sometimes have a non-negligible drop.

C. Year-on-year traffic increases

Since the goal of our model is to make predictions up to a year in advance, we would like to first observe whether there already is a year-to-year trend. In order to do so, we fit a linear regression to investigate whether trend for the number of vehicles over time exists (see fig. 4).

We first observe significant variance in the data. The number of vehicles has a standard deviation $\sigma \approx 270$, and data points are on average very far from the fitted trend (with $R^2 =$

0.029). Keeping this in mind, simply training a model that minimises loss is unlikely to generate predictions with such variance (as outputting the mean achieves the lowest average error). To combat this, we will incorporate a probabilistic component to our model to better replicate fluctuations.

Finally, the regression fit suggests an upwards trend in traffic over time with the number of vehicles climbing from approximately $10'000$ to $\sim 12'000$ per week. The gradient estimates an average increase of 3.353 vehicles per week over the course of our data.

III. Feature engineering

The raw data given to us is structured as a time series of discrete events. Each row represents a vehicle arrival and contains a timestamp of the arrival, the vehicle speed, and the vehicle weight. Other measurements such as lane of travel are discarded as they are not relevant for our main prediction task. We will reshape and expand these inputs and add new meaningful features to feed into our model.

A. Re-sampling

The first step in our model is to predict a rate of vehicles, so we transform our data from discrete events into an hourly arrival rate. We split the discrete data into intervals of 1h, so that our dataset is indexed by a fixed timestep of 1h over the whole range of time. We call this interval our *sampling interval*.

We count the number of vehicles in each sampling interval to create the `n_vehicles` feature. The speed and weight of the discrete events in the interval are also aggregated by a sum to create the total `speed` and `weight` features.

As a precaution, we filter out the dates past October 21st 2019 to avoid the unpredictable changes in traffic patterns due to the COVID-19 pandemic. We chose this exact date because it is the last non-zero data point until January 1st 2020, after which we consider the pandemic to have started.

B. HGV Driving restrictions

Heavy goods vehicles (*HGV*) in Switzerland are forbidden to drive on Sundays and on weekdays from 10 PM to 5 AM [3] . Since our data only includes vehicles over 3.5 tonnes, we need to take this into account. We add a binary `is_legal` feature whose value is 1 if it is legal for a HGV to drive and 0 otherwise.

C. Time periodicity

At the moment, the event timestamps are not very useful in their string representation. We would like to let our model consider any patterns in time, notably at the same hour of day (e.g. 7AM vs. 4PM), day of the week (e.g. Monday vs. Saturday), and time of the year (e.g. Christmas vs. summertime). Being traffic data, the rate of vehicles is clearly affected by these. In order to do so, we convert the timestamp into seconds, and apply a sine and cosine transform to obtain a periodic signal with respect to a wanted interval:

$$\textit{transform}(\text{timestamp}) = \begin{cases} \sin\left(\text{timestamp} \times \frac{2\pi}{L}\right) \\ \cos\left(\text{timestamp} \times \frac{2\pi}{L}\right) \end{cases}$$

where the *timestamp* input has already been converted into seconds and L is the interval length in seconds (e.g. for a day, $L = 60 * 60 * 24 = 86400$ seconds). One can uniquely reconstruct the date and time within a year from the transformed signals.

D. Splitting and normalising

We split our data into a training, validation and test dataset. The test and validation datasets contains two years of data each, the training set gets the remaining 4.4 years. Each year is considered to be 52 weeks long exactly in order to simplify windowing.

We standardise our features in each of the datasets by subtracting each feature by its mean and dividing it by its standard deviation in the training set (z-scoring).

E. Windowing

In order to create more data samples into our model, we use the concept of a sliding time window. We create a window object to generate our model inputs and outputs. This object takes a dataset and splits it into (input, output) tuples. It does so as follows:

1) It specifies a window of (1 year + 1 week). The input data will be the first week of the window, the output the last. This means that our model will predict a week using the same week of the previous year.
2) It takes the dataset and slides the window along the dataset, one hour at a time. Each slide creates a new (input week, output week) tuple.
3) It creates a new dataset from the generated tuples. The tuples are shuffled and batched into batches of 32. We feed the network one entire batch at a time. This is equivalent to performing stochastic gradient descent with minibatches, which reduces training time.

F. Zero weeks filtering

As seen in the exploratory data analysis section, our data comprises many weeks without any sensor input. If we were to include these datapoint in our rate prediction model training, it would introduce unwanted bias in our predictions. We need to filter these weeks out when training and evaluating performance.

The window object defined in the previous section also filters (input, output) tuples if either the input or the output number of vehicles is 0 for the whole week. It does so when transforming the data from from a Pandas DataFrame to a TensorFlow Dataset.

IV. Model structure and selection

A. Rate prediction

Our task is to use the hour-by-hour features described above from the input week to predict the vehicle count, sum of speeds, and sum of weights for our output week. We evaluated several possible models against our baseline to find the best performing model. We used mostly default configurations for our models.

Our baseline consists of outputting the same values from the previous week. We compare our models to it by computing the Mean Squared Error and Mean Absolute Error between the predicted week and the target week.

Our models are built using the Keras [4] API for Tensorflow [5]. They all use the default gradient descent algorithm provided by Keras and use early stopping evaluated on the validation dataset to avoid overfitting.

The models take a $H \times F$ matrix as input and output a $H \times N$ matrix. Here, $H = 168$ is the number of hours in a week, $F = 10$ is the number of input features, and $N = 3$ is the number of output features (`n_vehicles`, `speed`, `weight`).

We evaluated the following models:

1) *Smooth baseline*: A model similar to Baseline, but that tries to smooth out the stochasticity that comes from outputting the input. It takes the training data, and, for each hour of each week, computes the average values on all 4 years of the training data (discarding zero weeks).
2) *Linear model*: A linear regression model without regularisation
3) *Neural Network*: A neural network with a single hidden layer of 512 neurons and ReLU activation.
4) *LSTM*: A recurrent neural network with a layer of 32 LSTM cells, followed by a fully connected layer.

Once we find the best performing model, we predict the entire year of 2019 using the data from 2018 (see fig. 5). Both years are in our test set.

B. Discrete event generation

1) Vehicle arrival times: Once we have predicted a rate of vehicles passing in each time interval, we now have to generate discrete events. We will work upon the assumption that vehicle arrival times are drawn from a real-world stochastic process. Our aim is to draw samples from the same distribution to get random but representative results.

We can simulate such events with a Poisson process [6] with rate equal to the predicted vehicle rate. However, if one has a look at the given vehicle arrival data, the rate is not necessarily uniform over an hour-long time interval. Most notably, this is quite noticeable early in the morning and late in the evening, where one can respectively observe an "acceleration" and "deceleration" in vehicle arrival rates.

To account for this, we will assign a number of vehicles to smaller 5-minutes intervals within the hour, based on the probability distribution of a car belonging to the interval. These distributions are estimated for each hour of each day of the week based on previously observed data that was provided. Therefore, given n vehicles, we sample n times from the distribution to assign vehicles to the smaller intervals.

Finally, given n_i vehicles that have been assigned to interval i, one can generate arrival times with the following Poisson process with rate $\lambda = \frac{n_i}{5 * 60\text{sec}}$ (adapted from Prof. M. Bierlaire [6]):

1: $A \leftarrow$ empty list
2: $t = 0$
3: Draw $r \sim U(0, 1)$
4: $t = t - \ln(r)/\lambda$
5: **if** $t > T$ **return** A
6: A.append(t)
7: **GOTO** step 3

where $U \sim (0, 1)$ is the uniform distribution between 0 and 1, λ is the rate, and T is the 5-minute time length (300 seconds).

2) Weight and speed: For weight and speed predictions, we analogously sample from their corresponding distributions to produce representative observations to match with a given arrival time. From our data exploration, we observe that heavy vehicles above 60 tons are less likely to appear at night than during the day. Moreover, the variance in speed decreases as weight increases. Therefore, we accordingly sample vehicle weight based on the time of day, and speed based on the sampled weight.

V. Experimental results

In this section, we explore the different model possibilities and select the best one by using the rate prediction evaluation. We then assess the best interval samplings and discuss the performance of our discrete event generation.

A. Rate prediction performance

We evaluated the mean squared error and mean absolute error of our trained models on our test set. We also computed the relative difference with respect to our baseline for each metric. The results can be seen in Table I V-A.

Model	**MSE**	%	**MAE**	%
Baseline	0.3430	-	0.2959	-
Smooth	0.3435	+0.1	0.2959	0
Linear	0.3753	+9.43	0.4186	+41.47
Neural Net.	0.3273	−4.57	0.3596	+21.54
LSTM	0.2889	−15.78	0.3264	+10.32

TABLE I
MSE and MAE loss on test set for each model.
% is the relative difference to baseline

We can see that the linear model does not perform well. The relation between input and output is too complex to be modelled linearly. Both our neural nets outperform the smooth baseline model, which means that some additional behaviour is captured by our neural nets that is not just smoothing. All our models perform worse on mean absolute error. This metric is less important since small errors do not impact real world performance as much. The best performing model is our LSTM model, which outperforms our baseline by 15.8%.

B. Sampling interval selection

As discussed in the exploratory data analysis section, our data exhibits significant changes in arrival rates between hours. We therefore did not consider sampling intervals (the time step for which we count the number of vehicles) larger than 1h . It would make the second phase of our prediction model harder. However, we still wish to determine appropriate sampling intervals to predict vehicle arrival rates.

We also have to consider the trade-off between computation time and model performance. We evaluated our best performing model on a 1 hour, 30 min, and 10 min sampling interval. We also considered evaluating a 2 min interval, but its computation time was too great on our machines.

After evaluating our selected LSTM model on different sampling intervals, we obtained the following results:

Interval	1h	30 min	10 min
MSE (Val)	0.261	0.268	0.317

TABLE II
MSE OF LSTM FOR DIFFERENT SAMPLING RATES.
% IS THE RELATIVE DIFFERENCE TO BASELINE

We see that predictive performance decreases with smaller sampling intervals. We therefore choose to sample our weekly discrete data over a period of 1h. This strikes a good balance between the predictive power of our rate prediction model and the performance of our discrete generation phase, while also taking computation time into account. Moreover, increasing the sampling period beyond 1h would in theory decrease the rate prediction error. However we would see diminishing returns for the discrete decomposition as there is a higher loss of information concerning the period sub-intervals.

C. Discrete event generation performance

The quality of our generated discrete events highly depend on the initial predicted rate of vehicles, as it will define the random Poisson process that produces arrival times. By definition, in expectation we will produce the predicted number of vehicles.

As the attributes for the discrete events have been sampled/simulated in a probabilistic manner, there will be some variance as observed in the real-world, which is exactly what we wanted to emulate.

VI. FUTURE WORK

A. Not treating weeks as independent

We must acknowledge a limitation in our model that assumes week-to-week independence. At this stage it is unclear to us whether such trends exist (given the available data); however, our model will not adapt its predictions based on weekly changes. To remedy this issue, one could incorporate the week before prediction to account for short term fluctuations. However, this bars our ability to make a year-long prediction one year ahead of time. Nonetheless, one should consider adapting the model to perform shorter predictions for the immediate future. This would allow for more robust predictions given a sudden unexpected change in the traffic pattern.

B. Improved Poisson process

As it stands, arrival times are independently generated for each 5-minute time interval. However, this means that arrivals are less likely to appear at the start of these time frames. Instead of considering the intervals separately, one can construct a non-homogeneous Poisson process with a rate $\lambda(t)$ that varies over time (varies every 5-minutes to be exact). $\lambda(t)$ is equal to the rate of the interval that contains time t.

C. Accurate probability distributions

The probability distributions used to create event samples are constructed based on previously observed data. Although there is an abundance of datapoints for a given class, there is an insufficient amount of observations for other rare classes. For example, there are only 596 observations for vehicles above 80 tons. This accounts for only 0.0123% of total observations. The initial observation was that speed variance for high vehicle weights is smaller; however, this can simply be due to speeds far from the mean having low probability and therefore yield no representative samples in our observations. This means our estimated probability distributions are biased towards terms that are more likely to appear in previous observations. Here lies the big uncertainty within the problem as the actual distribution for rare events remains latent.

VII. CONCLUSION

The explored method generalises well to simulate traffic events a year in advance, assuming regular traffic patterns. One can easily adapt this model to short term predictions by considering smaller input-output windows, however the method remains the same. The observed real-world traffic events have large variance, therefore it is key to incorporate a stochastic event decomposition from predicted rates to emulate this behaviour.

REFERENCES

[1] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner, "Microscopic traffic simulation using sumo," in *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018. [Online]. Available: https://elib.dlr.de/124092/

[2] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 11 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735

[3] "Ordonnance sur les règles de la circulation routière - interdiction de circuler le dimanche et de nuit, art. 91," https://fedlex.data.admin.ch/filestore/fedlex.data.admin.ch/eli/cc/1962/1364_1409_1420/20210101/fr/pdf-a/fedlex-data-admin-ch-eli-cc-1962-1364_1409_1420-20210101-fr-pdf-a.pdf, January 2021.

[4] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[5] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[6] M. Bierlaire, "Simulating events: the poisson process," 2012. [Online]. Available: https://transp-or.epfl.ch/courses/OptSim2012/slides/05b-poisson.pdf

APPENDIX

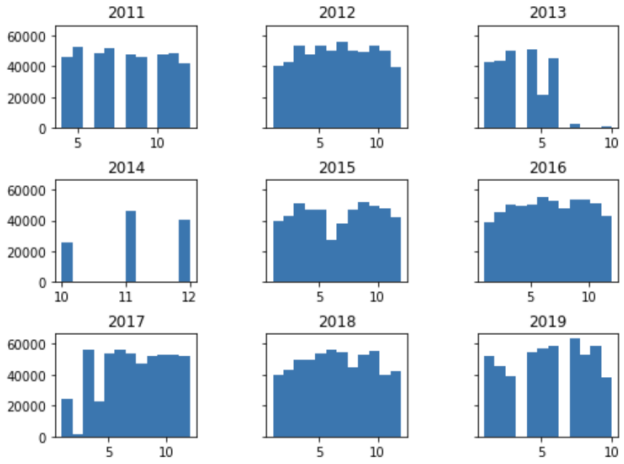

Fig. 1. Monthly number of vehicles per year. Horizontal axis corresponds to the month, the vertical axis to the number of vehicles per month.

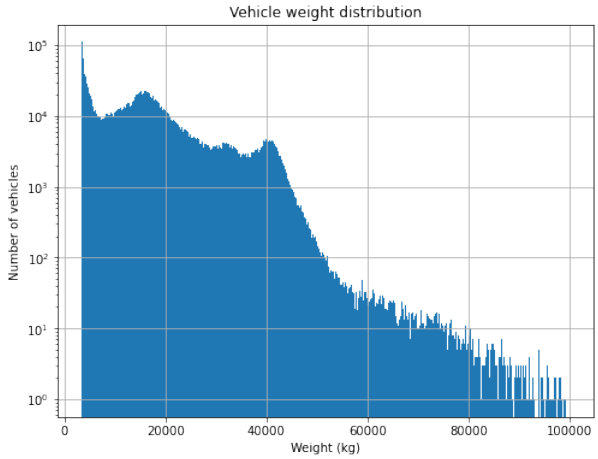

Fig. 2. Observed vehicle weight distribution

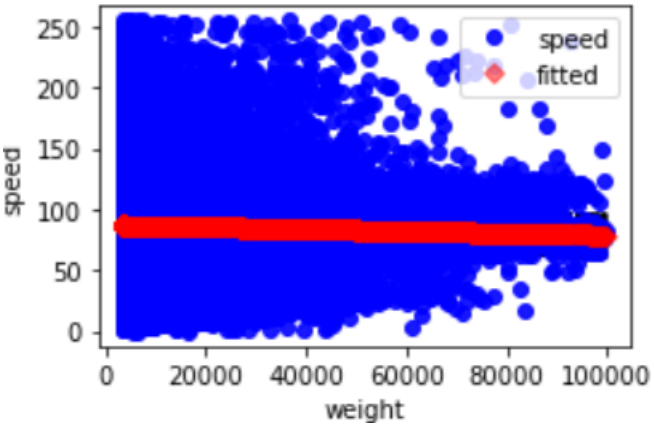

Fig. 3. Speed versus weight observations

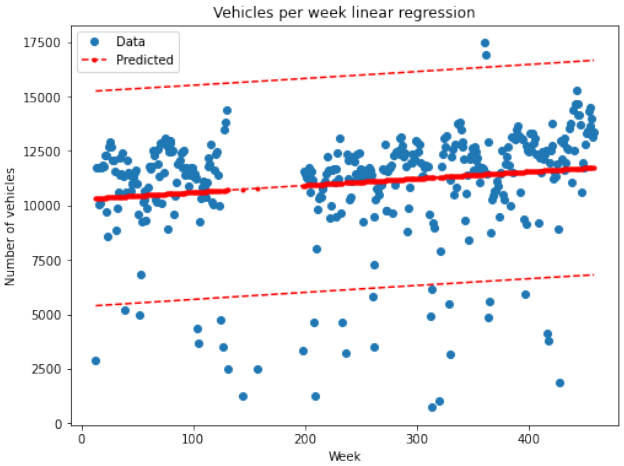

Fig. 4. Vehicles over time regression

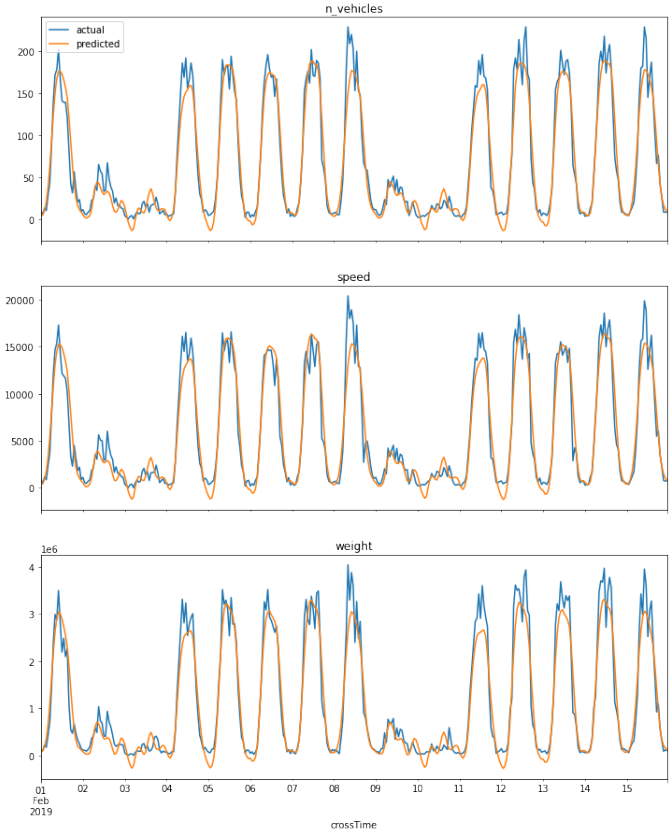

Fig. 5. Final predictions of columns versus the first 15 days of February